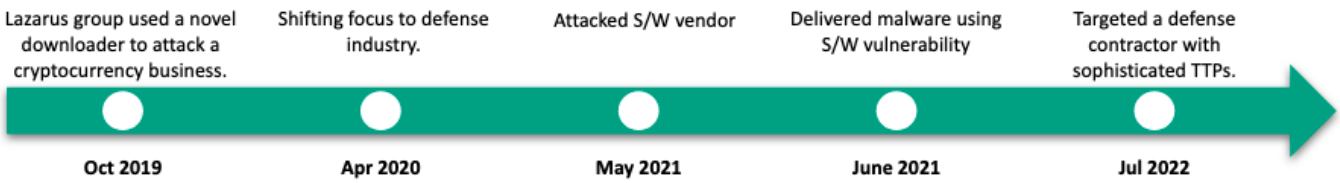
securelist.com /the-lazarus-group-deathnote-campaign/109490/

Following the Lazarus group by tracking DeathNote campaign

Authors

-  Seongsu Park

The Lazarus group is a high-profile Korean-speaking threat actor with multiple sub-campaigns. We have previously published information about the connections of each cluster of this group. In this blog, we'll focus on an active cluster that we dubbed DeathNote because the malware responsible for downloading additional payloads is named Dn.dll or Dn64.dll. This threat is also known as Operation DreamJob or NukeSped. Over the past few years, we have closely monitored the DeathNote cluster, observing a shift in their targets as well as the development and refinement of their tools, techniques, and procedures.

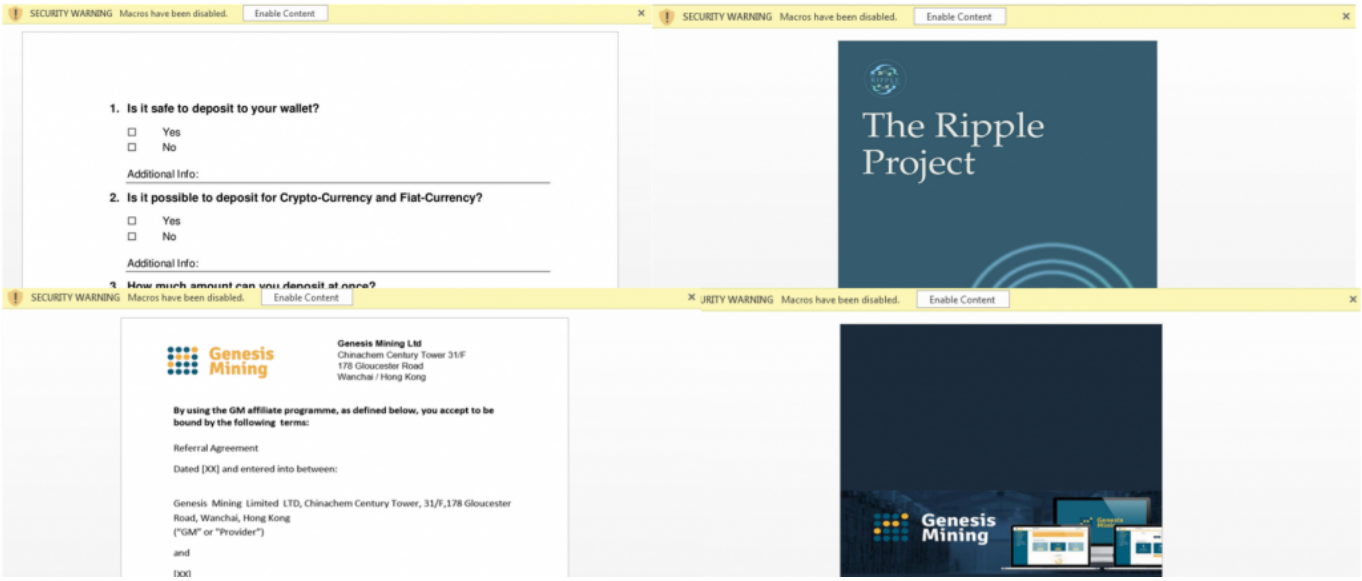


Timeline of DeathNote cluster

In this blog, we will provide an overview of the significant modifications that have taken place within this cluster, both in terms of its technical and strategic aspects.

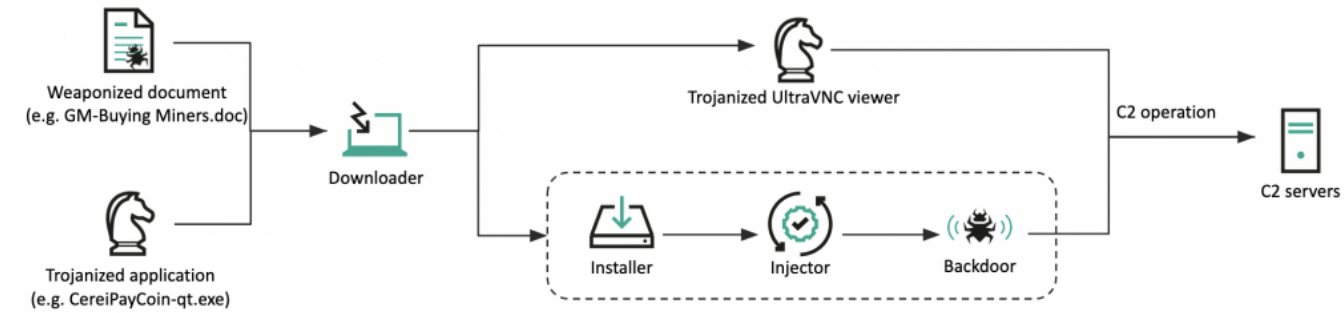
Beginning of tracking DeathNote

The notorious threat actor Lazarus has persistently targeted cryptocurrency-related businesses for a long time. While monitoring the actor's activities, we noticed that in one particular case they were using a significantly modified piece of malware. In mid-October 2019, we came across a suspicious document uploaded to VirusTotal. Upon further investigation, we discovered that the actor behind this weaponized document had been using similar malicious Word documents since October 2018. The malware author used decoy documents that were related to the cryptocurrency business such as a questionnaire about buying specific cryptocurrency, an introduction to a specific cryptocurrency, and an introduction to a bitcoin mining company.



Decoy documents

Once the victim opens the document and enables the macro, the malicious Visual Basic Script extracts the embedded downloader malware and loads it with specific parameters. In this initial discovery, the actor used two types of second-stage payload. The first is a manipulated piece of software that contains a malicious backdoor, while the second is a typical backdoor with a multi-stage binary infection process.



Infection procedure

The Trojanized application utilized in the second stage is masquerading as a genuine UltraVNC viewer. If executed without any command line parameters, it will display a legitimate UltraVNC viewer window. However, it carries out a malicious routine when it is spawned with "-s {F9BK1K0A-KQ9B-2PVH-5YKV-IY2JLT37QQCJ}" parameters. The other infection method executes the installer, which creates and registers an injector and backdoor in a Windows service. Finally, the backdoor is injected into a legitimate process (svchost.exe) and initiates a command-and-control (C2) operation. In this infection, the final

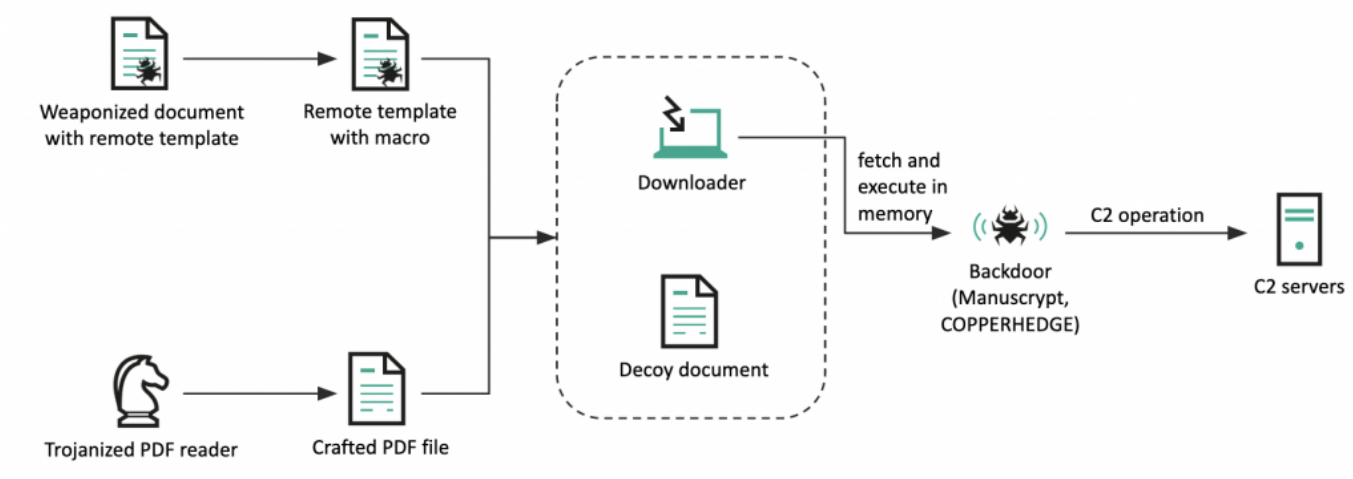
payload injected into the legitimate process was Manuscrypt. Until this discovery, the Lazarus group had primarily targeted the cryptocurrency business. Our investigation has identified potential compromises of individuals or companies involved in cryptocurrency in Cyprus, the United States, Taiwan and Hong Kong.

Shifting focus to the defense industry

While tracking this campaign, we uncovered a significant shift in the attack's target along with updated infection vectors in April 2020. Our research showed that the DeathNote cluster was used to target the automotive and academic sectors in Eastern Europe, both of which are connected to the defense industry. At this point, the actor switched all the decoy documents to job descriptions related to defense contractors and diplomatic services.

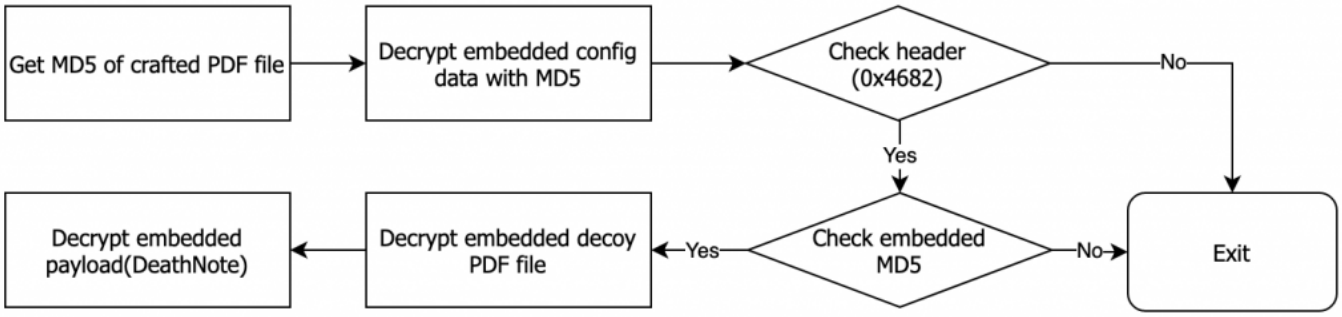
Decoy documents

In addition, the actor refined its infection chain, using the remote template injection technique in their weaponized documents, as well as utilizing Trojanized open-source PDF viewer software. Both of these infection methods result in the same malware (DeathNote downloader), which is responsible for uploading the victim's information and retrieving the next-stage payload at the C2's discretion. Finally, a COPPERHEDGE variant is executed in memory.



Infection chain

Notably, a Trojanized PDF reader, based on the open source software, used an interesting technique to initiate its malicious routine. It first retrieves the MD5 hash of the opened PDF file and performs an XOR operation on 65 bytes of embedded data using the retrieved MD5 value. Next, it verifies that the first WORD value of the XORed data is 0x4682, and checks that the MD5 hash value matches the last 16 bytes of the XORed data. If both conditions are met, the remaining 47-bytes value is used as the decryption key for the next stage of infection.



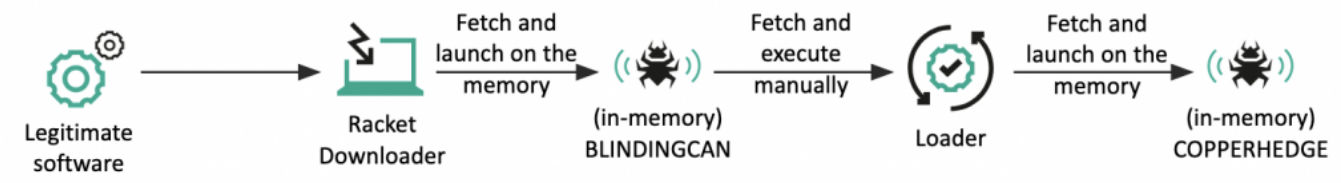
Verification process of Trojanized PDF reader

Finally, this Trojanized PDF viewer overwrites the original opened file with a decoy PDF file and opens it to deceive the victim while implementing the malware payload. The payload is executed with command line parameters, and a shortcut file is created in the Startup folder to ensure persistence. This infection mechanism demonstrates the care and precision with which the actor delivers the payload.

Expanded target and adoption of new infection vector

In May 2021, we observed that an IT company in Europe that provides solutions for monitoring network devices and servers was compromised by the same cluster. It's believed that the Lazarus group had an interest in this company's widely used software or its supply chain.

In addition, in early June 2021, the Lazarus group began utilizing a new infection mechanism against targets in South Korea. One thing that caught our attention was that the initial stage of the malware was executed by legitimate security software that is widely used in South Korea. It's thought that the malware was spread through a vulnerability in this widely used software in South Korea.

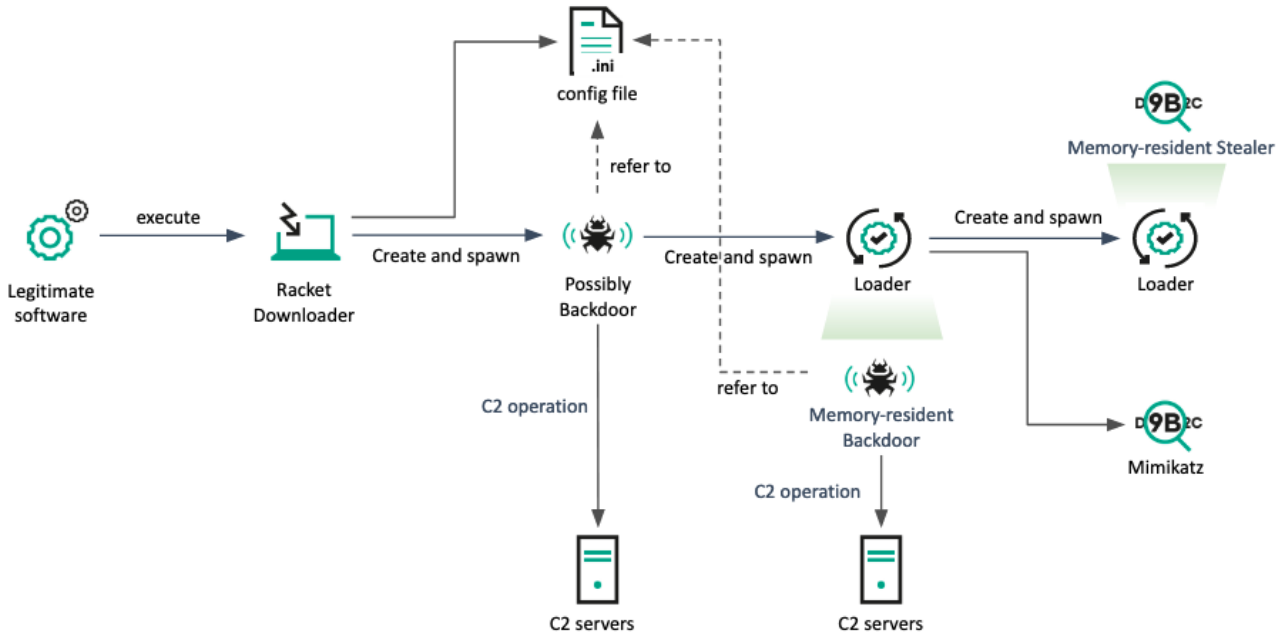


Infection chain

Similar to the previous case, the initial infection vector created the downloader malware. Once connected to the C2 server, the downloader retrieved an additional payload based on the operator's commands and executed it in memory. During this time, the BLINDINGCAN malware was used as a memory-resident backdoor. While the BLINDINGCAN malware has sufficient capabilities to control the victim, the actor manually implanted additional malware. It's believed that the group aims to create an auxiliary method to

control the victim. The retrieved loader's export function (CMS_ContentInfo) was launched with command line parameters, which is crucial for decrypting the embedded next-stage payload and configuration. This process only proceeds if the length of the parameter is 38. Finally, the COPPERHEDGE malware, previously used by this cluster, was executed on the victim.

Almost one year later, in March 2022, we discovered that the same security program had been exploited to propagate similar downloader malware to several victims in South Korea. However, a different payload was delivered in this case. The C2 operator manually implanted a backdoor twice, and although we were unable to acquire the initially implanted backdoor, we assume it is the same as the backdoor in the following stage. The newly implanted backdoor is capable of executing a retrieved payload with named-pipe communication. In addition, the actor utilized side-loading to execute Mimikatz and used stealer malware to collect keystroke and clipboard data from users.



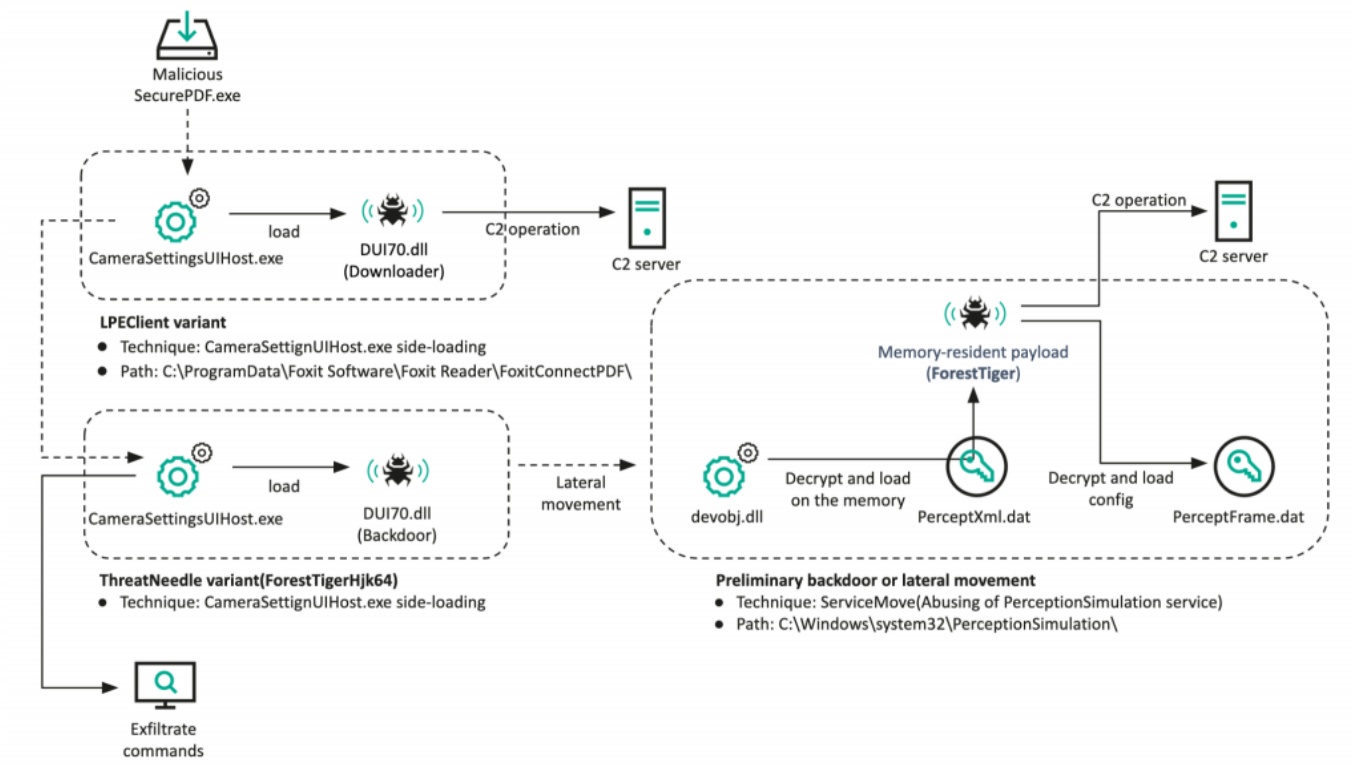
Infection chain

Around the same time, we uncovered evidence that one defense contractor in Latin America was compromised by the same backdoor. The initial infection vector was similar to what we've seen with other defense industry targets, involving the use of a Trojanized PDF reader with a crafted PDF file. However, in this particular case, the actor adopted a side-loading technique to execute the final payload. When the malicious PDF file is opened with the Trojanized PDF reader, the victim is presented with the same malware mentioned above, which is responsible for collecting and reporting the victim's information, retrieving commands and executing them using pipe communication mechanisms. The actor used this malware to implant additional payloads, including legitimate files for side-loading purposes.

- Legitimate file: %APPDATA%\USOShared\CameraSettingsUIHost.exe
- Malicious file: %APPDATA%\USOShared\dui70.dll
- Config file: %APPDATA%\USOShared\4800-84dc-063a6a41c5c
- Command line: %APPDATA%\USOShared\CameraSettingsUIHost.exe uTYNkfKxHiZrx3KJ

An ongoing attack targeting a defense contractor with updated infection tactics

In July 2022, we observed that the Lazarus group had successfully breached a defense contractor in Africa. The initial infection was a suspicious PDF application, which had been sent via the Skype messenger. After executing the PDF reader, it created both a legitimate file (CameraSettingsUIHost.exe) and malicious file (DUI70.dll) in the same directory. This attack heavily relied on the same DLL side-loading technique that we observed in the previous case. The payload that was initially implanted and executed by the PDF reader was responsible for collecting and reporting the victim's information, as well as retrieving an additional payload from the remote server named LPEClient. The Lazarus group used this malware several times in various campaigns. They have also utilized the same DLL side-loading technique to implant additional malware that is capable of backdoor operation. In order to move laterally across systems, the actor used an interesting technique called ServiceMove. This technique leverages the Windows Perception Simulation Service to load arbitrary DLL files. According to the author's explanation, 'a non-existing DLL file will be loaded every time when the Windows Perception Simulation Service is started'. By creating an arbitrary DLL in C:\Windows\System32\PerceptionSimulation\ and starting the service remotely, the actors were able to achieve code execution as NT AUTHORITY\SYSTEM on a remote system. The actor created a devobj.dll file in the PerceptionSimulation folder and remotely executed the PerceptionSimulation service. Upon launching the devobj.dll file, it decrypted an encrypted backdoor file, PercepXml.dat, from the same folder and executed it in memory.



Infection chain

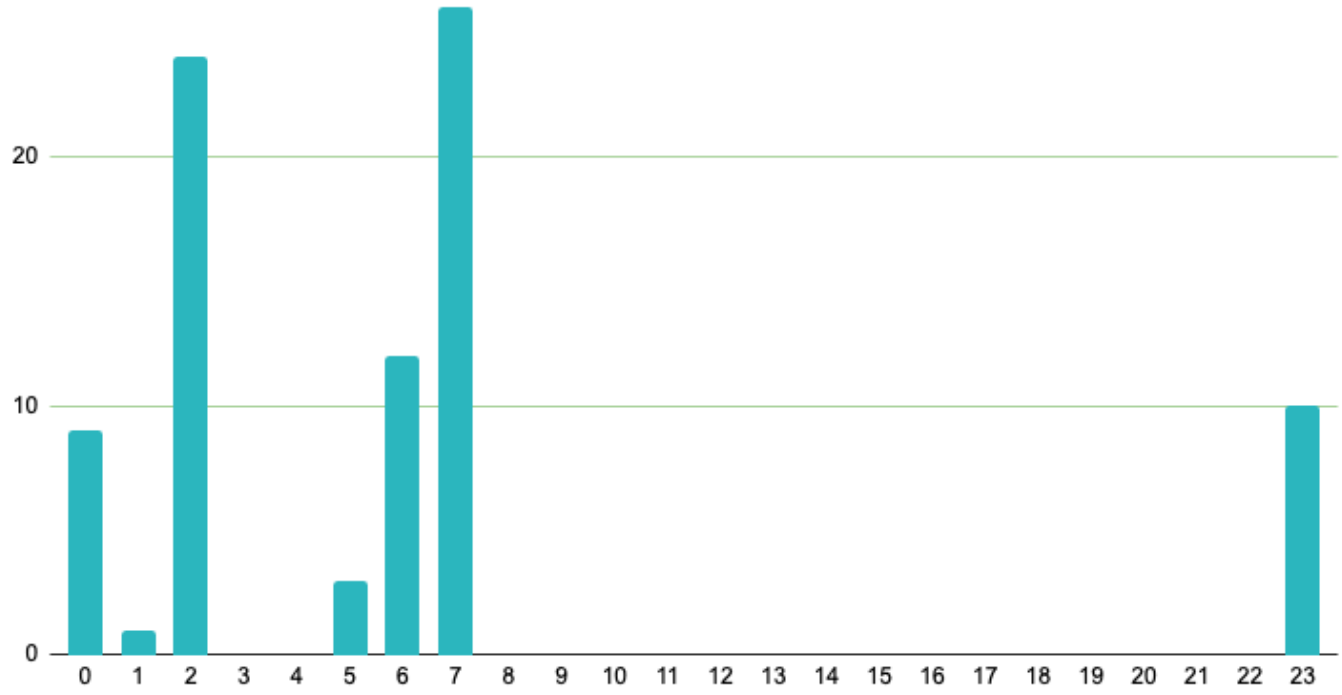
Post-exploitation

During our investigation of this campaign, we have gained extensive insight into the Lazarus group's post-exploitation strategy. After initial infection, the operator executed numerous Windows commands to gather basic system information and attempt to find valuable hosts, such as an Active Directory server. Before moving laterally, the Lazarus group acquired Windows credentials using well-known methods, and employed public techniques, such as ServiceMove. When the group completed its mission and began exfiltrating data, they mostly utilized the WinRAR utility to compress files and transmit them via C2 communication channels.

Phase	Examples
Basic reconnaissance	Generally used Windows commands. For example: • cmd.exe /c netstat -ano | find TCP • systeminfo In one case, they accessed the default domain controllers policy directly. • cmd.exe /c "Type "\\[redacted]\SYSVOL\[redacted]\Policies\{6AC1786C-016F-11D2-945F-00C04fB984F9}\MACHINE\Microsoft\Windows NT\SecEdit\GptTmpl.inf"
Finding high-value hosts	To find a connected Remote Desktop host it utilized Windows commands or queried the saved server list from the registry. • cmd.exe /c netstat -ano | findstr 3389 • cmd.exe /c reg query HKEY_USERS\S-1-5-*[redacted]*-1001\Software\Microsoft\Terminal Server Client\Servers Utilizing ADFind tool to acquire Active directory information. • cmd.exe /c "%appdata%\[redacted].xic -b dc=[redacted],dc=[redacted] -f "sAMAccountName=[redacted]" >> %temp%\dm3349.tmp 2>&1″
Acquiring login credentials	Utilizing crafted Mimikatz to dump login credentials or Responder tool to capture credentials.
Lateral movement	One common approach for launching commands on remote hosts is to use methods like SMB connection or the ServiceMove technique.
Exfiltration	Using WinRAR to archive files before sending the stolen file via C2 channel. • adobearm.exe a -hp1q2w3e4 -m5 -v2000000k "%Local AppData%\Adobe\SYSVOL800.CHK" "\\[redacted]FILE02.[redacted]\Projects\[redacted] Concept Demonstrator" • %appdata%\USOShared\USOShared.LOG1 a -hpb61de03de6e0451e834db6f185522bff -m5 "%appdata%\USOShared\USOShared.LOG2" "%appdata%\ntuser.001.dat"

Attribution

After tracking the DeathNote cluster and its origin, we have determined that the Lazarus group is responsible for this malware strain. Our conclusion is supported by many security vendors who also believe that the Lazarus group is linked to this malware. Furthermore, we have analyzed the delivery of Windows commands to the victim through the DeathNote malware, and discovered that a significant number of commands were executed between GMT 00:00 and 07:00. Based on our knowledge of normal working hours, we can infer that the actor is located in either the GMT+08 or GMT+09 time zone.



Timeline of Windows commands

Moreover, the actor left a Korean comment '정상호출', which translates to 'normal call' in the C2 script. This further supports the hypothesis that Lazarus is a Korean-speaking actor.

```
If Request.QueryString("productid") = "9405" Then
    If Request.QueryString("num") = "8927345" Then ' 정상호출
```

Korean comment in the C2 script

In conclusion, the Lazarus group is a notorious and highly skilled threat actor. Our analysis of the DeathNote cluster reveals a rapid evolution in its tactics, techniques and procedures over the years. As the Lazarus group continues to refine its approaches, it is crucial for organizations to maintain vigilance and take proactive measures to defend against its malicious activities. By staying informed and implementing strong security measures, organizations can reduce the risk of falling victim to this dangerous adversary.

Indicators of Compromise

Beginning of tracking DeathNote

Malicious documents

265f407a157ab0ed017dd18cae0352ae
7a73a2261e20bdb8d24a4fb252801db7
7a307c57ec33a23ce9b5c84659f133cc
ced38b728470c63abcf4db013b09cff7
9121f1c13955506e33894ffd780940cd
50b2154de64724a2a930904354b5d77d
8a05f6b3f1eb25bcbceb717aa49999cd
ee73a772b72a5f3393d4bf577fc48efe

Downloader

d1c652b4192857cb08907f0ba1790976
25b37c971fd7e9e50e45691aa86e5f0a
0493f40628995ae1b7e3ffacd675ba5f
8840f6d2175683c7ed8ac2333c78451a
c278d6468896af3699e058786a8c3d62
9fd35bad075c2c70678c65c788b91bc3
59cb8474930ae7ea45b626443e01b66d
7af59d16cfd0802144795ca496e8111c
cd5357d1045948ba62710ad8128ae282
77194024294f4fd7a4011737861cce3c
e9d89d1364bd73327e266d673d6c8acf
0d4bdfec1e657d6c6260c42ffdbb8cab
5da86adeec6ce4556f477d9795e73e90
706e55af384e1d8483d2748107cbd57c

Manipulated Installer

dd185e2bb02b21e59fb958a4e12689a7

Installer

4088946632e75498d9c478da782aa880 C:\Windows\igfxmon.exe

Injector

dc9244206e72a04d30eeadef23713778 C:\Windows\system32\[random 2 bytes]proc.exe

Backdoor

735afcd0f6821cbd3a2db510ea8feb22 C:\Windows\system32\[random 2 bytes]svc.dll

Shifting focus to the defense industry

Malicious documents

4c239a926676087e31d82e79e838ced1 pubmaterial.docx
183ad96b931733ad37bb627a958837db Boeing_PMS.docx
9ea365c1714eb500e5f4a749a3ed0fe7 Boeing_DSS_SE.docx
2449f61195e39f6264d4244dfa1d1613 Senior_Design_Engineer.docx
880b263b4fd5de0ae6224189ea611023 LM_IFG_536R.docx.docx

e7aa0237fc3db67a96ebd877806a2c88 Boeing_AERO_GS.docx
56470e113479eacda081c2eeead153bf boeing_spectrolab.docx

Fetched template
2efbe6901fc3f479bc32aaf13ce8cf12 pubmaterial.dotm
65df11dea0c1d0f0304b376787e65ccb 43.dotm
0071b20d27a24ae1e474145b8efc9718 17.dotm
1f254dd0b85edd7e11339681979e3ad6 61.dotm

DeathNote downloader
f4b55da7870e9ecd5f3f565f40490996 onenote.db, thumbnail.db
2b02465b65024336a9e15d7f34c1f5d9 wsuser.db
11fdc0be9d85b4ff1faf5ca33cc272ed onenote.db
f6d6f3580160cd29b285edf7d0c647ce
78d42cedb0c012c62ef5be620c200d43 wsuser.db
92657b98c2b4ee4e8fa1b83921003c74
075fba0c098d86d9f22b8ea8c3033207 wsdts.db
8fc7b0764541225e5505fa93a7376df4
7d204793e75bb49d857bf4dbc60792d3 2.dll
eb2dc282ad3ab29c1853d4f6d09bec4f
ca6658852480c70118feba12eb1be880 thumbnail.db
c0a8483b836efdbae190cc069129d5c3 wsdts.db
14d79cd918b4f610c1a6d43cadeeff7b wsuser.db
1bd0ca304cdecfa3bd4342b261285a72

Trojanized PDF viewer
cbc559ea38d940bf0b8307761ee4d67b SumatraPDF.exe
da1dc5d41de5f241cabd7f79fbc407f5 internal pdf viewer.exe

Expanded target and adoption of new infection vector

Racket Downloader
b3a8c88297daecdb9b0ac54a3c107797 SCSKAppLink.dll

BLIDINGCAN
b23b0de308e55cbf14179d59adee5fcb
64e5acf43613cd10e96174f36cb1d680

COPPERHEDGE Loader
a43bdc197d6a273102e90cdc0983b0b9

COPPERHEDGE
97336f5ce811d76b28e23280fa7320b5

Downloader Loader
f821ca4672851f02bead3c4bd23bed84 c:\officecache\officecert.ocx

Racket Downloader
b974bc9e6f375f301ae2f75d1e8b6783 %public%\Libraries\SCSKAppLink.dll
eb061dfacb3667cf65d250911179235d

Stealer
fe549a0185813e4e624104d857f9277b %ProgramData%\GenICam\GenICamKDR.gic

Backdoor Loader
7b8960e2a22c8321789f107a7b83aa59 %ProgramData%\xilinx\xilinx.pkg
0ac90c7ad1be57f705e3c42380cbcccd %ProgramData%\USOShared\USOShare.cpl

Mimikatz Loader
adf0d4bbefccf342493e02538155e611 %ProgramData%\USOShared\log.dll
d4d654c1b27ab90d2af8585052c77f33

An ongoing attack targeting a defense contractor with updated infection tactics

Loader
2bcf464a333d67afeb80360da4dfd5bb C:\Windows\system32\perceptionsimulation\devobj.dll
83dd9b600ed33682aa21f038380a6eab C:\Windows\system32\perceptionsimulation\devobj.dll

ForestTiger(Backdoor)
97524091ac21c327bc783fa5ffe9cd66 ProgramData\adobe\arm\lockhostingframework.dll
9b09ebf52660a9d6deca21965ce52ca1 %appdata%\adobe\arm\DUI70.dll
26c0f0ce33f5088754d88a1db1e6c4a9

Trojanized PDF reader
84cd4d896748e2d52e2e22d1a4b9ee46 SecurePDF.exe